International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:10, 2007

Strategic Software Development: Productivity Comparisons of General Development Programs

Craig Comstock, Zhizhong Jiang, and Peter Naudé

Abstract**—**Productivity has been one of the major concerns with the increasingly high cost of software development. Choosing the right development language with high productivity is one approach to reduce development costs. Working on the large database with 4106 projects ever developed, we found the factors significant to productivity. After the removal of the effects of other factors on productivity, we compare the productivity differences of the ten general development programs. The study supports the fact that fourth-generation languages are more productive than third-generation languages.

Keywords**—**Functional point, Language, Productivity, Software Engineering.

I. Introduction

OVER the past years dramatic improvements in hardware performance, profound changes in computing architectures, and vast increases in memory and storage capacity have precipitated more sophisticated and complex computer-based systems [1]. Software is the key element in the evolution of computer-based systems and products. While hardware costs have decreased considerably comprising less than one fifth of total expenditure, the cost of software remains consistently high [2]. One of the primary problems in software development that have yet to be solved satisfactorily is making systems cost effective. A major obstacle to solve the problem of cost effective is the intrinsic complexity in developing software. Improving the productivity is an essential part of making system cost effective [3].

The problem of productivity associated with cost deserves our serious attention. Previous studies have focused in great part on the discovery of methods and identification of factors for productivity improvement [4-10]. With the increasing complexities and costs of software development, how to improve development productivity has been an ongoing concern for project managers.

Manuscript received June 5, 2007. This research was supported by International Software Benchmarking Standards Group (ISBSG).

Craig Comstock was with Harvard University. He is now with University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD UK (e-mail: craig.comstock@lmh.ox.ac.uk).

Zhizhong Jiang was with Department of Statistics, University of Oxford. He is now with University of Manchester, Booth Street West, Manchester, M15 6PB, UK (phone:+44(0)8708328157; fax: +44(0)1612756596; e-mail: zhizhong.jiang@bnc.oxon.org).

Peter Naudé is a Professor in Manchester Business School, University of Manchester, Booth Street West, Manchester, M15 6PB, UK (e-mail: pete.naude@mbs.ac.uk).

In retrospect of the past studies in software engineering there have been few comparing the productivity levels of various development programs. The main reason is the lack of accessible and reliable large dataset [11]. Besides, many contemporary metrics repositories have limited use due to their obsolescence and ambiguity of documentation [12].

The data repository maintained by the International Software Benchmarking Standards Group (ISBSG) does not have the above deficiencies and has been widely researched [11, 13-16]. Focusing on the statistical analysis of the latest release of ISBSG data repository with 4106 projects, this paper compares the productivity levels of ten common development programs. Project coordinators can adopt the findings of this paper by choosing the most productive programs suitable for their development.

The paper is organized as follow. Section II gives an overview of the development programs that are in common use. Section III briefly introduces the main software metrics or information involved in the analysis. Section IV and V are the detailed procedures of model development and validation. Section VI presents the comparisons of the ten development programs regarding productivity. Finally, section VII is the conclusion of this study.

II. Overview of General Development Programs

In the past, over a thousand different programming languages have been designed by various groups and international committees [17]. Whereas a large number of programs were superseded, there are still many remained in current use. In the ISBSG data repository, the common development programs that were frequently used are C, C++, COBOL, Java, PL/1, SQL, Visual Basic, PowerBuilder, Oracle and Access. Although some other programming languages (e.g., Delphi, C#) have also been broadly used in practice, they were not included in our analysis due to their lack of popularity in the data repository. We now briefly introduce the ten development programs.

1) C

Originated as a systems programming language, C combines the advantages of a high-level language with the facilities and efficiency of an assembly language [17]. As a typical procedural language, C has spread its use in diverse areas and is regarded as a general-purpose language.

2) C++

As an extension of C, C++ was designed to be an efficient and practical language. It is one of the primary object-oriented language and remains extremely popular for non-web applications [18].

3) COBOL

As the dominant programming language for business application, COBOL (Common Business-Oriented Language) has been widely applied in the past. Its main deficiency is that complex algorithms are extremely difficult to program in COBOL [18].

4) Java

As a common object-oriented language, Java has the real virtues of being relatively simple, cleanly designed and easily portable. It is currently being used not only for Internet and network applications, but also for general applications [18].

5) PL/1

Though it is unpopular today, PL/1 is of significant historical importance for its contribution to the programming language design and development methods [19]. It was designed with the objective of combining all the best features of FORTRAN and COBOL [18].

6) SQL

SQL (Structured Query Language) is a query language that enables database programmers to retrieve or modify data in most relational databases. Literally hundreds of database products now support SQL which stands today as the standard computer database language [20].

7) Visual Basic

Visual Basic is an event-driven programming language and has its object-oriented features [19]. It allows programmers to easily create simple GUI applications, and also has the flexibility to develop fairly complex applications.

8) PowerBuilder

PowerBuilder has the object-oriented power of 3GL along with the GUI feature. It distinguishes from other languages for its ability to handle large-scale projects and its open systems approach [21]. With its own scripting language PowerScript, it is used primarily for building business applications.

9) Oracle

Oracle is a relational database management system. Its family of database products includes several powerful applications development and generation tools. These tools can efficiently conduct the work of database management, data access and manipulation, programming, and connectivity [22].

10) Access

Access is a powerful database package and development tool that has established itself as a standard for database management [23]. Its main strengths are the speed and facility to develop database related applications.

III. Data Description

The data repository has one parameter Primary Programming Language which describes the development program used for the specific project. Although this parameter was recorded with nominal scale, we cannot use simple parametric or other nonparametric tests to compare the differences of productivity for the development programs. The reason is before comparing group differences we have to remove or control the influences of other factors [24]. That is, before making comparisons of different development programs, the effects of other factors on productivity have to be considered. Based on the attributes of all the underlying factors significant to productivity, we applied multiple regression analysis. We now introduce the software metrics or descriptive pieces of information recorded in the data repository which are related to our study.

(1) Normalized Productivity Delivery Rate (PDR)

PDR is the parameter which directly measures the level of productivity. It is calculated from Normalized Work Effort divided by Adjusted Function Points. Normalized Work Effort represents the effort in total hours for the development, and Adjusted Function Points is the measure of project size. Clearly, PDR is an inverse measure of productivity in that the larger PDR, the smaller is the productivity.

(2) Average Team Size

It is the average number of people that worked on the project through the entire development process. Past studies suggest that productivity and team size are negatively associated [10, 25-27].

(3) Primary Programming Language

It specifies which programming language was used for the development (e.g., C++, Java).

(4) Development Type

It describes whether software development was a new development, enhancement or re-development. It has been suggested that development with enhancement may consume much of the total resources of programming groups, and therefore does not necessary improve productivity [28].

(5) Development Platform

It defines the primary platform for the development. The project was developed for one of the platforms of Mid-range platform, Mainframe, Multi or personal computer (PC). Subramanian et al. [29] found platform has a significant effect on software development effort. This may indicate this factor is likely to influence development productivity.

(6) Development Techniques

These are the techniques used during software development (e.g. Waterfall, Prototyping, Data Modeling etc). A large number of projects adopted joint uses of different techniques. Among the various development techniques, Rapid Application Development (RAD) was reported to significantly accelerate development [30].

(7) Case Tool Used

It indicates whether the project used any CASE (Computer-Aided Software Engineering) tool. While some studies reported CASE tool had a positive effect on productivity [31-33], many organizations responded that it has not brought about a change in productivity [34]. Bruckhaus et al. [35] pointed out that the introduction of CASE tool does not necessarily improve productivity, and in certain situations it can actually decrease the productivity as it increases effort on specific activities.

(8) How Methodology Acquired

It describes how the development methodology was acquired. It can be Traditional, Purchased, Developed In-house, or a combination of Purchased and Developed. Liu and Mintram [11] found development methodology is not significant to effort, which is one of the determinants of productivity.

(9) Data Quality Rating

It indicates the reliability of the data recorded. It has four grades A, B, C, and D. While the data with quality ratings A, B and C are assessed as being acceptable, little credibility can be given to any data with rating D.

It is important to point out that that some scholar regarded project duration as an important factor for productivity, and productivity declines with project duration increasing [10]. However, we did not take this factor into account as our study is to explore the factors that intrinsically influence productivity. In fact, project duration is correlated with effort which is one of the two determining elements of productivity.

IV. Model Development

We first validate the data before model development. To have robust results we excluded those projects with rating D of data quality, since little credibility should be given to them. Besides, projects with recording errors or unspecified information were removed. For instance, two projects were mistakenly recorded with Average Team Size 0.5 and 0.95 respectively.

Second, we examine if there exists the problem of multicollinearity (strong correlations between predictor variables) in the data. That is, to see whether the use of some development method is likely to be associated with other techniques. The correlation tests indicated that there is no multicollinearity existent in the data.

Finally, for the metric Development Techniques there exist over 30 different techniques in the data repository. Our research focused on the ten primary techniques, and separated each of the ten techniques as one single binary variable with two levels indicating whether it was used or not (1 = used, 0 = not used). These ten techniques are Waterfall, Data Modelling, Process Modelling, JAD (Joint Application Development, Prototyping, Regression Testing, Object Oriented Analysis & Design, Business Area Modelling, RAD (Rapid Application Development), and Event Modelling. Given that many projects adopted various forms of joint uses of different techniques, we did not consider the interplay of these techniques. For all the uncommon development techniques, they were merged into one group labelled with 'Others'. Table I below generalizes the variables for the analysis.

TABLE I
Descriptions of the Variables in the Analysis

Variable	Scale	Descriptions
PDR	Ratio	Normalized Productivity Delivery Rate.
TeamSize	Ratio	Average Team Size.
Language	Nominal	Primary Programming Language
DevType	Nominal	Development Type
Platform	Nominal	Development Platform
CASE	Nominal	CASE Tool Used
Methodology	Nominal	How Methodology Acquired
Waterfall	Nominal	1= Waterfall, 0 = Not
Data	Nominal	1 = Data Modelling, 0 = Not
Process	Nominal	1 = Process Modelling, 0 = Not
JAD	Nominal	1 = JAD, 0 = Not
Regression	Nominal	1 = Regression Testing, 0 = Not
Prototyping	Nominal	1 = Prototyping, 0 = Not
Business	Nominal	1 = Business Area Modelling, 0 = Not
Event	Nominal	1 = Event Modelling, 0 = Not
RAD	Nominal	1 = Rapid Application Development 0 = Not
OO	Nominal	1 = Object Oriented Analysis & Design 0 = Not
Others	Nominal	1 = uncommon development techniques 0 = Not

Table I showed that PDR and TeamSize are the only two variables measured in ratio scale. We now examine their distributions with histogram in Fig. 1 below. It displayed that the data are highly skewed. Therefore, log-transformations were applied to them (see Fig. 2).

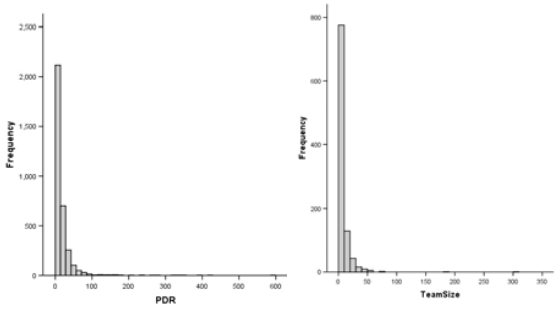

Fig. 1 Histograms of PDR and TeamSize

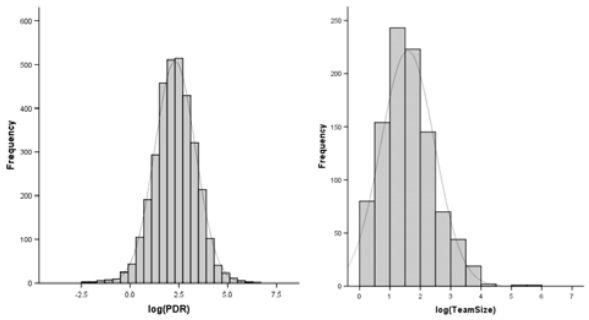

Fig. 2 Histograms of PDR and TeamSize with log-transformation

Fig. 3 below is the scatterplot of log(PDR) against log(TeamSize). Whereas they do not have a perfect positive linear relationship, the graph indicates that we can use linear model to approximate their relationship. Given that all other predictors are measured in nominal scale except TeamSize, we can use multiple linear regression to fit a model with PDR as the dependent variable.

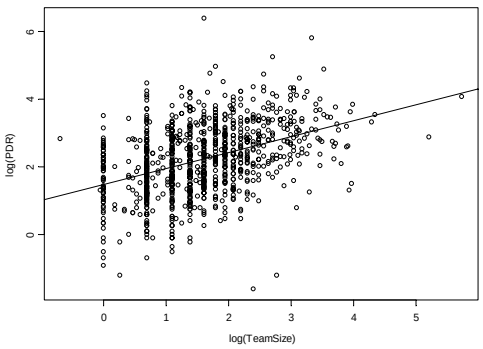

Fig. 3 The scatter plot of log (PDR) against log(TeamSize)

For multiple regression analysis, the rule of thumb suggests a minimum sample size of 50+8k (k is the number of predictors) [36]. Although there exist considerable missing values in the data, the valid sample size is 330 after data cleaning. This is sufficient to perform regression analysis. In statistical package S-plus, we conducted multiple linear regression with the core data. The resultant model contains the predicator variables that are significant to the dependent variable based on the normal criterion of significance (p-value <0.05). The model was fitted as:

$$\log(PDR)$$
$$= 1.058 + 0.337 \times \log(TeamSize) + \alpha_i I(Language_i) + \beta_j I(Platform_j)$$
$$i = 0,2,..10,\ j = 0,1,2,3$$

Some interpretations are necessary to understand the model.

1) *PDR* is Normalized Productivity Delivery Rate; *TeamSize* is Average Team Size for the development; $Language_i$ represents one of the development languages used by the project; $Platform_j$ represents one of the four platforms used for the development. α_i and β_j are the regression coefficients. Table II shows the summary of the regression results.
2) log( ) is the natural log with base e. The indicator function $I(\cdot)$ outputs only two values: value of 1 means the relevant technique in the parentheses is used, where value of 0 indicates not (That is, $I(a)$=1 if and only if a is used).

TABLE II
SUMMARY OF THE REGRESSION ANALYSIS

	Terms	Coefficients
	Intercept	1.058
	log(TeamSize)	0.337
i	$Language_i$	α_i
0	Access	0
1	C	1.558
2	C++	1.127
3	COBOL	1.300
4	Java	1.169
5	ORACLE	0.807
6	PL/1	0.655
7	PowerBuilder	0.908
8	SQL	1.053
9	Visual Basic	0.921
10	Other	0.827
j	$Platform_j$	β_j
0	Mainframe	0
1	Mid-range	-0.440
2	Multi	-0.592
3	PC	-0.634

V. MODEL VALIDATION

The model fitted is parsimonious with the minimum number of predictors significant to productivity. While the saturated model contains all the predictors and has the most perfect goodness-of-fit, our parsimonious model was reported with multiple R^2 of 0.402. This indicates the fitted model is acceptable with 40.2% of the variance in the dependent variable explained by the minimum number of predictors.

Furthermore, in linear model it is assumed that the residuals are normally distributed with zero mean and homogeneity of variance [37]. Equal scatter of residual points about the horizontal axis indicates the residuals have homogeneity of variance [38]. Fig. 4 below is the diagnostic plot of residuals against fitted values. It shows that the residual points evenly scatter along the horizontal axis without obvious patterns. Therefore, the assumption of homogenous variance is validated.

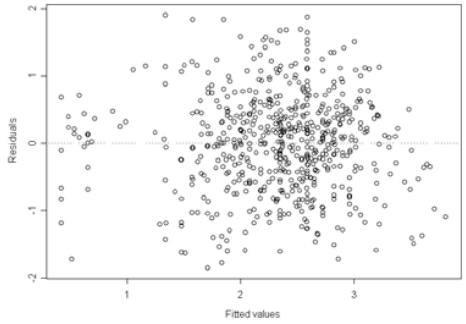

Fig. 4 Diagnostic plot of residuals against the fitted values

Finally, we applied Quantile-Quantile plot to check the assumption of normality of the residuals. The approximately straight line in Fig. 5 indicates that the residuals do not deviate from normal distribution.

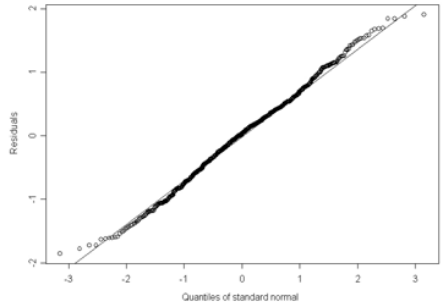

Fig. 5 Quantile-Quantile plot of the residuals

Therefore the fitted model is feasible. We now turn to the discussions of the model and compare the productivity differences of the ten development programs.

VI. Productivity Differences of Development Programs

The fitted model includes the predictor variables that are significant to the dependent variable. In other words, these predictors are the factors that influence productivity, these being Average Team Size, Primary Programming Language, and Development Platform.

In section III we mentioned PDR is an inverse measure of productivity in that the smaller PDR, the higher is the productivity. Based on the fitted model we can see that Average Team Size and productivity are negatively associated. Particularly the double of Average Team Size will reduce productivity by 20% (1-exp(-0.337*ln2)). The finding of negative effect of team size on productivity is consistent with past studies [10, 25-27].

For the two significant factors Primary Programming Language and Development Platform, the smaller value of the coefficients of the function $I(\cdot)$ leads to smaller value of log(PDR), indicating higher productivity of the corresponding development approaches. Therefore, projects that were developed for platforms Multi and PC have higher productivity than those developed for platforms Mainframe and Mid-range.

After controlling the effects of team size and development platform on productivity, we can compare the differences of productivity among the ten common development languages. The comparisons are based on their coefficients of the indicator function $I(\cdot)$ in Table II with Access acted as the reference language. The findings are generalized as follows.

(1) Visual Basic (0.921), Power Builder (0.908), SQL (1.053), Oracle (0.807), and Access (0) are more productive than C (1.558), C++ (1.127), Java (1.169), and COBOL (1.300). The first five belong to fourth-generation languages (4GL) and the left belong to third-generation languages (3GL). In practice 4GLs are designed to reduce programming efforts, and they are more productive than 3GLs [39].

(2) With the smallest coefficient of the indicator function $I(\cdot)$, Access is the most productive development program. One of its main strengths is the speed in which database-related applications can be developed. As the leading database management system, Oracle achieves reasonably high productivity.

(3) The two most prevailing development languages C++ and Java have comparable productivity (the coefficients are 1.127 and 1.169 respectively).

(4) As a traditional procedural language, C has the lowest productivity. C has been widely criticized for its difficulty to achieve effective operations. The effective use of C requires more experience and effort than other programming languages. As the mostly widely used language in business area, COBOL has the second lowest productivity. Although PI/1 is moderately productive, it is considered to be outdated in software industry.

VII. Conclusion

This study worked on the latest release of ISBSG data repository which is deemed as an influential database for the study of software metrics. Multiple regression analysis was conducted with the core data. To compare the productivity differences of the ten common development programs, we removed the effect of other significant factors on productivity, these being Average Team Size and Development Platform. The results support the fact that fourth-generation languages are more productive than third-generation languages. By comparing the coefficients of the regression terms, we found Access has the highest productivity while C language has the lowest. As the two prevailing development languages, C++ and Java have the same level of productivity. Whereas Oracle has reasonably high productivity, the productivity of COBOL is low.

REFERENCES

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. London: McGraw-Hill, 2005.
[2] H. V. Vliet, *Software Engineering: Principles and Practice*. Chichester: Wiley, 1993.
[3] S. T. Albin, *The Art of Software Architecture: Design Methods and Techniques*. New York: Wiley, 2003.
[4] J. D. Blackburn, G. D. Scudder, and L. N. V. Wassenhove, "Improving speed and productivity of software development: a global survey of software developers," *IEEE Transactions on Software Engineering*, vol. 22, pp. 875-885, 1996.
[5] B. W. Boehm and P. N. Papaccio, "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering*, vol. 14, pp. 1462-1477, 1988.
[6] D. N. Card, F. E. McGarry, and G. T. Page, "Evaluating software engineering technologies," *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 845-851, 1987.
[7] G. R. Finnie, G. E. Wittig, and D. Petkov, "Prioritizing software development productivity factors using the analytic hierarchy process," *Journal of Systems and Software*, vol. 22, pp. 129-139, 1993.
[8] N. R. Howes, "Managing software development projects for maximum productivity," *IEEE Transactions on Software Engineering*, vol. SE10, pp. 27-35, 1984.
[9] R. E. Loesh, "Improving productivity through standard design templates," *Data Processing*, vol. 27, pp. 57-59, 1985.
[10] K. Maxwell, L. V. Wassenhove, and S. Dutta, "Software development productivity of European space, military and industrial applications," *IEEE Transactions on Software Engineering*, vol. 22, pp. 706-718, 1996.
[11] Q. Liu and R. C. Mintram, "Preliminary data analysis methods in software estimation," *Software Quality Journal*, vol. 13, pp. 91-115, 2005.
[12] W. Harrison, "A flexible method for maintaining software metrics data: a universal metrics repository," *Journal of Systems and Software*, vol. 72, pp. 225-234, 2004.
[13] C. J. Lokan, "An empirical analysis of function point adjustment factors," *Information and Software Technology*, vol. 42, pp. 649-660, 2000.
[14] R. Jeffery, M. Ruhe, and I. Wieczorek, "A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data," *Information and Software Technology*, vol. 42, pp. 1009-1016, 2000.
[15] J. J. Cuadrado-Gallego, M. Sicilia, M. Garre, and D. Rodríguez, "An empirical study of process-related attributes in segmented software cost-estimation relationships," *Journal of Systems and Software*, vol. 79, pp. 353-361, 2006.
[16] J. Moses, M. Farrow, N. Parrington, and P. Smith, "A productivity benchmarking case study using Bayesian credible intervals," *Software Quality Journal*, vol. 14, pp. 37-52, 2006.
[17] L. B. Wilson and R. G. Clark, *Comparative Programming Languages*. Wokingham: Addison-Wesley, 1988.
[18] K. C. Louden, *Programming Languages: Principles and Practice* London: Brooks/Cole, 2003.
[19] R. Cezzar, *A Guide to Programming Languages: Overview and Comparison*. Boston: Artech House, 1995.
[20] J. R. Groff and P. N. Weinberg, *SQL: The Complete Reference New* York McGraw-Hill, 2002.
[21] D. D. Deyhimi, D. S. Heath, and D. Mosley, *Advanced PowerBuilder 4.0 Techniques*. New York: Wiley, 1995.
[22] J. T. Perry, *Understanding Oracle*. San Francisco: Sybex, 1989.
[23] E. Jones, *Developing Client/Server Applications with Microsoft Access*. London: McGraw-Hill, 1997.
[24] R. R. Newton and K. E. Rudestam, *Your Statistical Consultant: Answers to Your Data Analysis Questions*. London: SAGE, 1999.
[25] F. Louis, "Team size and productivity in systems development," *Information Systems Management*, vol. 8, pp. 27-35, 1991.
[26] S. D. Conte, H. E. Dunsmore, and Y. E. Shen, *Software Engineering Metrics and Models*. Redwood City, CA: Benjamin-Cummings Publishing, 1986.
[27] E. Mendes and B. Kitchenham, "Web Productivity Measurement and Benchmarking," in *Web Engineering*, E. Mendes and N. Mosley, Eds. Berlin: Springer, 2006, pp. 75-106.
[28] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, "Characteristics of application software maintenance," *Communications of the ACM*, vol. 21, pp. 466-471, 1978.
[29] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," *Empirical Software Engineering*, vol. 11, pp. 541-553, 2006.
[30] J. Martin, *Rapid Application Development*. New York: Macmillan, 1991.
[31] R. D. Banker and R. J. Kauffman, "Reuse and productivity in integrated computer-aided software engineering: an empirical study," *MIS Quarterly*, vol. 15, pp. 375-401, 1991.
[32] C. Necco, N. W. Tsai, and K. W. Holgeson, "Current usage of CASE software," *Journal of Systems Management*, vol. 40, pp. 6-11, 1989.
[33] R. T. Coupe and N. M. Onodu, "An empirical evaluation of the impact of CASE on developer productivity and software quality," *Journal of Information Technology*, vol. 11, pp. 173-181, 1996.
[34] D. Flynn, J. Vagner, and O. D. Vecchio, "Is CASE technology improving quality and productivity in software development?" *Logistics Information Management*, vol. 8, pp. 8-23, 1995.
[35] T. Bruckhaus, N. H. Madhavii, I. Janssen, and J. Henshaw, "The impact of tools on software productivity," *IEEE Software*, vol. 13, pp. 29-38, 1996.
[36] S. A. Green, "How many subjects does it take to do a multiple regression analysis?" *Multivariate Behavioral Research*, vol. 26, pp. 499-510, 1991.
[37] A. C. Rencher, *Linear Models in Statistics*. New York: John Wiley & Sons, 2000.
[38] W. J. Krzanowski, *An Introduction to Statistical Modelling*. London: Arnold, 1998.
[39] R. Klepper and D. Bock, "Third and fourth generation language productivity differences," *Communications of the ACM*, vol. 38, pp. 69-79, 1995.

Craig Comstock is a PhD student at the University of Oxford. He received his B.E. from Harvard University (Cum Laude), MSc in Software Engineering from University of Oxford,, and MBA (with highest honors) from University of Chicago.

Zhizhong Jiang is a PhD student at Manchester Business School, University of Manchester, United Kingdom. He received his B.E.(first-class honors) from Harbin Institute of Technology (China) and MSc in Applied Statistics from University of Oxford.

Peter Naudé is a Professor of Marketing at Manchester Business School, University of Manchester, United Kingdom. He publishes widely in business studies and information systems.